LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-214595

Tracking Vehicles in traffic Surveillance Video

M. Maire, C. Kamath

August 15, 2005

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Tracking Vehicles in Traffic Surveillance Video

Michael Maire

Department of Electrical Engineering and Computer Science

University of California, Berkeley

mmaire@eecs.berkeley.edu

Chandrika Kamath

Center for Applied and Scientific Computing

Lawrence Livermore National Laboratory

kamath2@llnl.gov

Abstract

We present a system for detecting and tracking vehicles in surveillance video. Our algorithm uses a simple motion model to determine salient regions in a sequence of video frames. Similar regions are associated between frames and clustered to yield coherent final tracks. The entire process is automatic and uses computation time that scales according to the size of the input video sequence.

Keywords: Urban surveillance video, Tracking, Saliency.

1 Introduction

Automatically detecting and tracking vehicles in video surveillance data is a challenging problem in computer vision with important practical applications, such as traffic analysis and security. Video cameras are a relatively inexpensive surveillance tool. However, manually reviewing the large amount of data they generate is often impractical. Thus, algorithms for analyzing video which require little or no human input are an attractive solution and have been an area of active research for over a decade. In addition to correctness in detecting and tracking vehicles, the computational complexity of a tracking system is important. For many applications, real-time or near real-time tracking capability is desired.

Our approach is designed with these requirements in mind. The computational complexity of our algorithm is linear in the size of a video frame and the number of vehicles tracked. The algorithm differs from some of the previous vehicle tracking work in that it does not require any user calibration of road or camera location. Instead, we use low-level cues to pick out moving regions, from which we build template-based appearance models for vehicles. These models are used to associate the detections of a vehicle in different frames, yielding a coherent track.

In the remainder of the paper, we first discuss related tracking research in Section 2. Next, we present the main components of our tracking algorithm, the background model, appearance templates, and clustering procedure, in Section 3. In Section 4 we display results on several traffic surveillance videos. We conclude in Section 5 with a discussion of our algorithm's performance and suggestions for future improvements.

2 Related Work

Vehicle tracking has a long history in the computer vision literature. Koller et. al. [4, 5, 6] have described a few different early approaches. In [4], parameterized 3D polyhedral vehicle models are matched to coherently moving image features. This algorithm uses an offline camera calibration step to aid in recovery of the 3D pose. An iterated extended Kalman Filter is used to update estimates of the model's location and pose.

In [5] and [6], the authors use an adaptive background model to estimate the location of

moving blobs in a scene, which are tracked using a Kalman filter based motion model. They use an explicit occlusion reasoning step in order to disambiguate overlapping vehicles. This step is dependent on knowledge of the scene geometry, as the authors assume motion is constrained to the ground plane. A final contribution of this work is the use of belief networks to describe high-level events in the scene, such as lane changes and stalled vehicles. Computing this type of information from the individual vehicle tracks is a useful surveillance tool.

The authors of [1] focus on obtaining real-time performance from a vehicle tracking system. Their algorithm finds and tracks corner features in video and then groups the resulting individual feature tracks into vehicles using common motion as a cue. Taking advantage of a user specified road geometry, they group neighboring features whose depth-corrected relative displacement remains consistent with vehicle motion. In order to achieve real-time performance on the hardware available in 1997, the authors implemented their system on specialized digital signal processors.

More recently, [3] applies a feature tracking approach to traffic viewed from a low-angle off-axis camera. Vehicle occlusions and perspective effects pose a more significant challenge for a camera placed low to the ground. To confront this challenge, their system uses a pre-specified scene geometry to judge vehicle height, which is used as a cue for grouping features.

As background models are integral to a number of the above approaches, recent work has also focused on developing sophisticated and reliable background models. In [2], the authors compare the performance of a large set of different background models on urban traffic video. They also experiment with sequences filmed in weather conditions such as snow and fog, for which a robust background model is required.

3 Tracking Algorithm

Our general tracking approach is to extract salient regions from the video using a learned background model, and build template models for vehicle appearance from these regions. We first apply a local motion model to the difference between consecutive frames to produce a map of salient foreground pixels. The foreground is segmented into regions which are used as templates for a normalized correlation based tracker. Computing the similarity between each

template and its spatially nearby possible matches in the next frame, we have a criterion by which to group regions into coherent objects. Extraneous tracks that violate the appearance or motion characteristics of a vehicle are then dropped from the final output. We discuss the details of each of these steps in the sections below.

3.1 Background Model

A thorough survey of background models for urban traffic surveillance is presented in [2]. As our approach is not centered on background modeling, we implement a simple background model that is effective on the videos of interest. Our background model is used as an initial processing stage and could easily be swapped for more sophisticated or computationally intensive methods or those tailored to specific weather conditions if the need arose. The desired output of this stage is simply a binary map indicating which image pixels belong to the foreground, as shown below in Figure 1(b).

(a) (b) (c)

Figure 1: (a) Probabilistic output of the background model, (b) thresholded foreground map, and (c) resulting foreground regions outlined.

To generate this map, we use the fact that a vehicle is a group of pixels that move in a coherent manner, either a lighter region over a dark background or vice versa. More specifically, while the vehicle may be the same color as the background, at least some portion of it must appear different from at least one part of the background against which it is seen (unless it is perfectly camouflaged, in which case no vision system can detect it). We ignore any vehicle that doesn't move under the intuition that any vehicle that is of interest to track will move at least once during the surveillance video (especially over long periods of surveillance). It is then

trivial to go back and identify the presence of a stationary vehicle if the track begins or ends within the scene.

We model the difference between consecutive intensity values at the same image pixel as a Gaussian, $I_{t+1}(x,y) - I_t(x,y) \sim N(\mu, \sigma)$, where μ and σ are set to the sample mean and standard deviation of the intensity differences observed over a fixed number of surrounding frames. Note that μ and σ are independent of the pixel location (although a location-dependent model is also possible). In addition, I_t and I_{t+1} are normalized images, with the intuition that we want to correct for any sudden changes in lighting, such as may be caused by the sun emerging from behind a cloud. Though we do not attempt to correct for localized illumination changes, one could do so by performing a per pixel or per region normalization based on a larger surrounding region.

As an initial step, we smooth the difference images in both the spatial and temporal dimensions before learning or applying the background model. We use Gaussian kernels with a standard deviation of 1 pixel in each spatial dimension and 2 frames in the temporal dimension. Since vehicles are large blocks of pixels moving in the same manner, smoothing reduces noise while leaving the signal from true vehicles untouched.

Figure 1 shows the results of this process. Figure 1(a) displays, for each location, the probability of a lower intensity difference than that observed given that the pixel in question is part of the background. We threshold this probability to obtain the foreground map in Figure 1(b). The threshold was chosen in order to produce few false positive foreground detections.

For the experiments discussed later in Section 4, we used the entire video sequence to estimate the parameters of the background model. However, a real-time system could dynamically update these parameters with constant cost per frame by simply keeping a buffer of a fixed number of frames.

3.2 Appearance Templates

Segmenting the foreground map into connected components and taking the convex hull of each, we obtain the foreground regions shown in Figure 1(c). The entire region within the convex hull is used since vehicles tend to be convex and in addition, their interiors might be too uniform

to be picked up by the motion model (as is the case with the upper interior of the bus in Figure 1(b)).

Figure 2: Templates for foreground components. Shown here are the templates generated for the ten largest foreground regions displayed above in Figure 1(c).

For each foreground region, we take its surrounding image patch as a template for use in matching its appearance in the surrounding frames. Our code enables either the entire rectangular patch surrounding the region, or only the portion of the image patch within the convex region, to serve as a template. Using the entire patch captures some of the background in the template, but may be useful if the background changes slowly and there is a strong boundary between the vehicle and background. The experiments reported later use the full image patch.

Figure 2 demonstrates the templates obtained for the video frame from Figure 1. We track each template by computing its best normalized correlation score in the next frame. This is equivalent to convolving normalized versions of the template and succeeding frame. Figure 3 illustrates the result for one template. In practice, we do not search the entire image for the best normalized correlation score, but rather just a local window, since we can place an upper bound on the maximum vehicle velocity (a more sophisticated scheme could also estimate the velocity from the data, but for our purposes it suffices to set this parameter by hand). To handle vehicles of different scales, we set the search window side length to be twice that of the vehicle template.

(a) template (b) image (c) correlation

Figure 3: Normalized correlation response (c) on image (b) for the template shown in (a). The template in (a) was extracted from the image 10 frames prior to that shown in (b). The best match for the template, indicated by the dark red dot, correctly identifies the car in question. The second best match is a different white car that is similar in appearance.

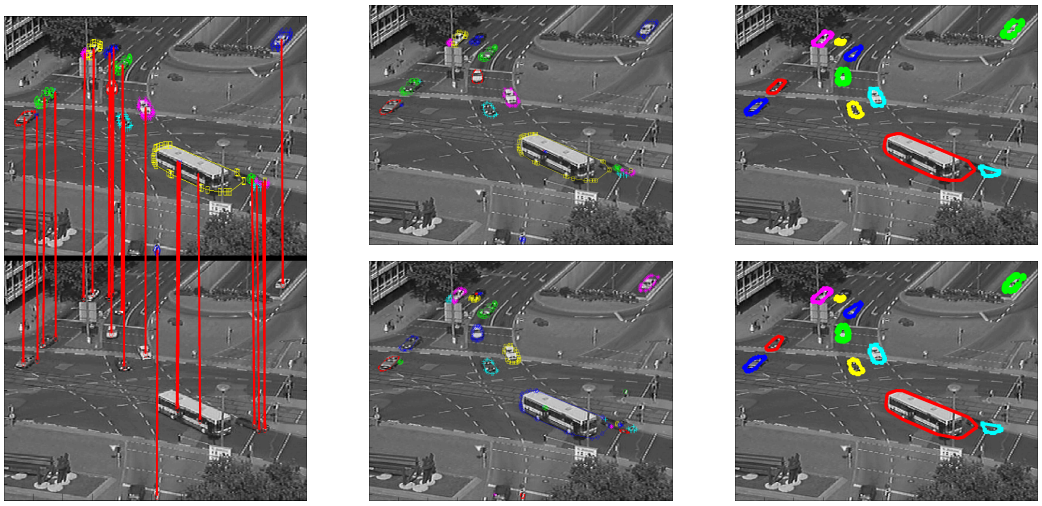
3.3 Clustering Tracks

By using the normalized correlation tracker to compute a template's best match in both the preceding and succeeding frames, we have an estimate of the position of the object represented by the template in those frames. Unifying regions with significant overlap in their tracked locations produces a single track for each vehicle, as shown in Figure 4. We also drop small regions that have not been unified with larger regions as they are likely due to noise from the background model.

4 Results

We tested our algorithm on publicly available surveillance video of traffic intersections obtained from [8]. Typical results are shown in Figures 5 and 6.

Our unoptimized Matlab implementation runs at a speed of approximately 2 seconds per frame on somewhat dated hardware (a 1.5GHz Pentium IV). The background modeling stage is relatively fast, as it simply involves a fixed number of operations per pixel. In addition, the template matching process is parallelizable, so in combination with further optimization or custom hardware, a real-time implementation is feasible. Moreover, initial experiments indicate



(a) (b) (c)

Figure 4: Foreground components in consecutive frames are clustered based on the output of the normalized correlation tracker. (a) Lines indicate the best match of each region from the top frame in the next consecutive frame (shown on the bottom). (b) All regions initially identified in the two frames. (c) We produce a consistent vehicle track by clustering regions from different frames based on the proximity of their matches.

graceful degradation of the system when run on lower resolution video to improve processing speed.

5 Discussion and Future Work

While we obtained decent results on a number of test video sequences, there are many areas in which we can look to make future improvements.

As previously mentioned, in the future it could be advantageous to plug in a more sophisticated background model, capable of dealing with weather conditions such as fog or snow. Swapping out the background model presented here for a different one is straightforward and does not affect the operation of the rest of the processing pipeline.

Integration of an explicit occlusion reasoning phase, possibly as a post-processing step, could repair the instances in which our algorithm incorrectly split one vehicle track into two as a result

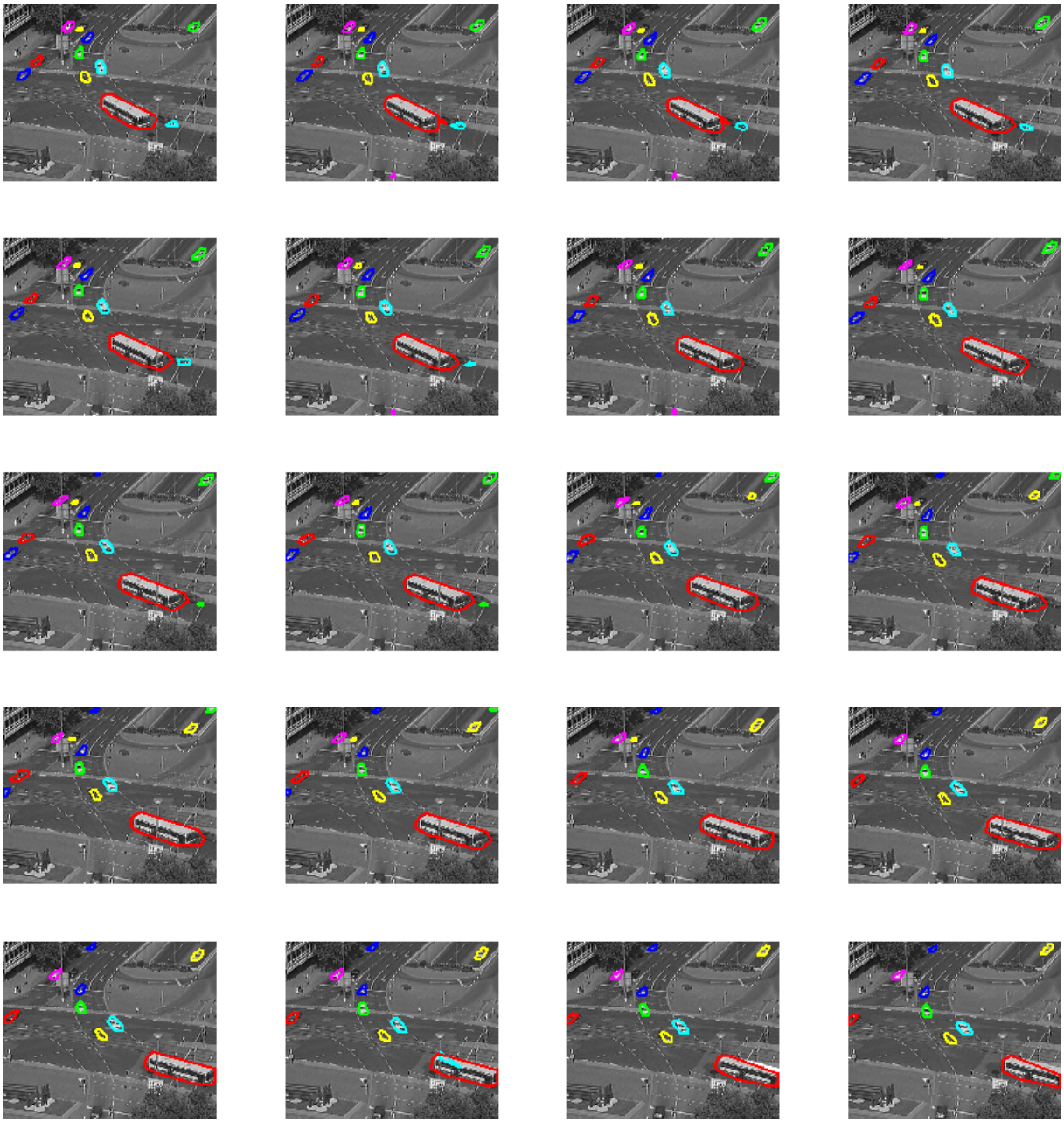


Figure 5: Vehicle tracking results for surveillance video of a traffic intersection. Cars are color-coded according to their identities. Every other frame of a 40 frame sequence is displayed.

of an occlusion. Figure 6 displays a few examples of this situation. Another valuable tool in correcting tracker errors could be the use of road and traffic flow models. There is existing work on automatic road extraction from video [7].

Finally, it may be possible to use the model of each vehicle's appearance that we recover as part of the tracking process in novel ways. For example, it could be used to detect the same vehicle in novel scenes, such as one monitored by a different traffic camera, or search for a detection of a specific class of vehicle in a video database.

Acknowledgments: We would like to thank the KOGS/IAKS Universität Karlsruhe for providing their video data for public use.

This research was performed while Michael Maire was on appointment as a U.S. Department of Homeland Security (DHS) Fellow under the DHS Scholarship and Fellowship Program, a program administered by the Oak Ridge Institute for Science and Education (ORISE) for DHS through an interagency agreement with the U.S Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORISE.

UCRL-TR-xxxxxx. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, *A Real-time Computer Vision System for Measuring Traffic Parameters*, IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[2] S. Cheung and C. Kamath, *Robust Techniques for Background Subtraction in Urban Traffic Video*, Video Communications and Image Processing, SPIE Electronic Imaging, San Jose, January 2004.

[3] N. Kanhere, S. Pundlik, S. Birchfield, *Vehicle Segmentation and Tracking from a Low-Angle Off-Axis Camera*, IEEE Conference on Computer Vision and Pattern Recognition, San Diego, June 2005.

[4] D. Koller, *Moving Object Recognition and Classification based on Recursive Shape Parameter Estimation*, In Proc. of the 12th Israeli Conf. on Artificial Intelligence, Computer Vision, and Neural Networks, pp. 359-368, Tel-Aviv, Israel, December 27-28, 1993.

[5] D. Koller, J. Weber, T. Haung, J. Malik, G. Ogasawara, B. Rao, and S. Russell, *Towards Robust Automatic Traffic Scene Analysis in Real-Time*, In Proc. of the 12th Int'l Conference on Pattern Recognition (ICPR-94), pp. 126-131, Jerusalem, Israel, October 9-13, 1994.

[6] D. Koller, J. Weber, and J. Malik, *Robust Multiple Car Tracking with Occlusion Reasoning*, In Proc. Third European Conference on Computer Vision, LNCS 800, Springer-Verlag, 1994.

[7] R. Pless and D. Jurgens, *Road Extraction from Motion Cues in Aerial Video*, Proceedings of the ACM Conference on Geographic Information Systems, 2004.

[8] KOGS/IAKS Universität Karlsruhe. Available at: http://i21www.ira.uka.de/image_sequences/

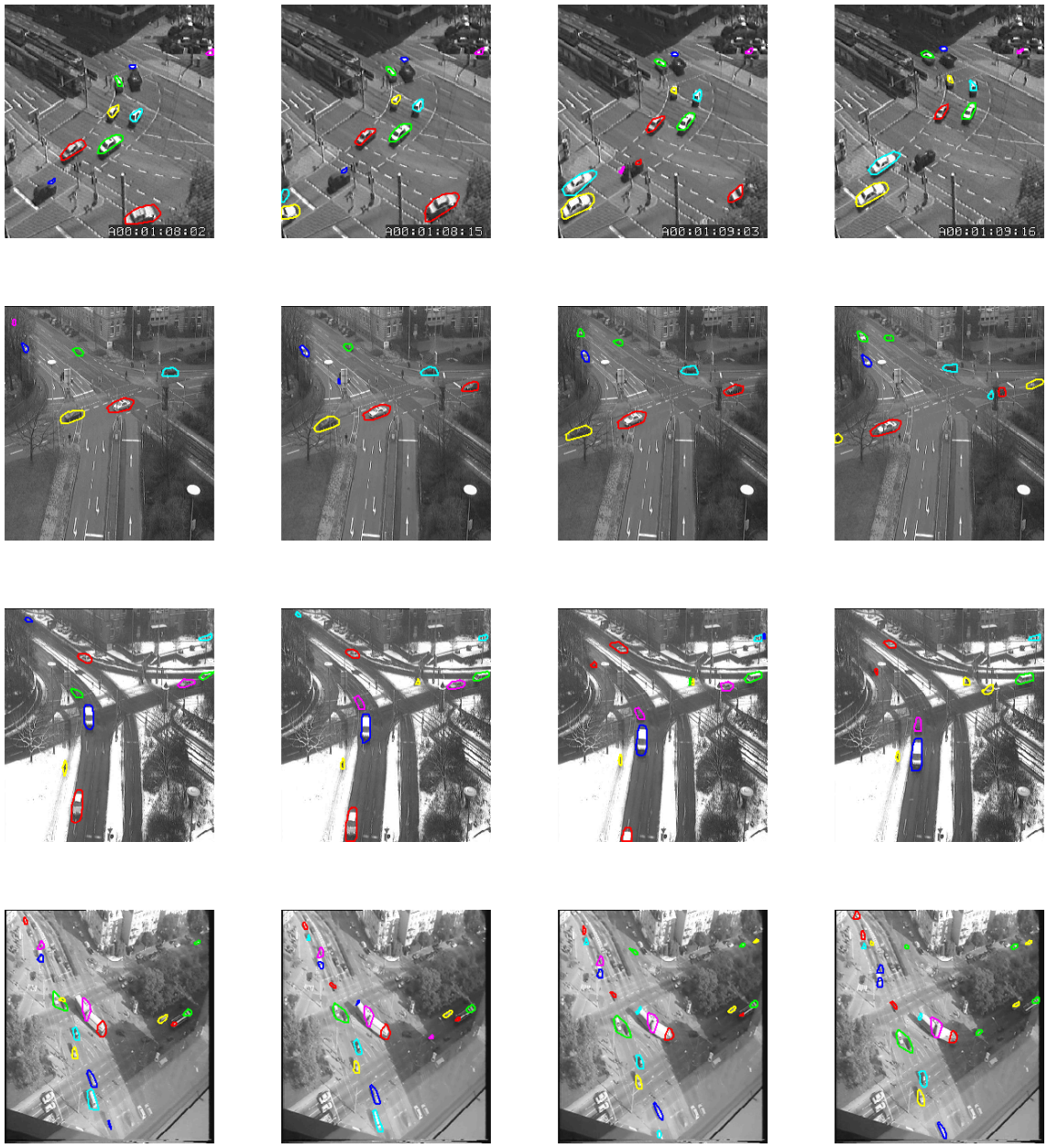


Figure 6: Tracking results on four different traffic video sequences. A 40-frame window of each sequence is shown above. We ran the same algorithm on each sequence without altering any parameter settings. The majority of cars are correctly identified and tracked, however lower resolution cars (bottom row) are particularly challenging. Our background model also has difficulty with the double trailer in this row. Some occlusions are correctly handled (lower right of top row and upper left of second row), while others could benefit from explicit occlusion reasoning as a post processing step (right side of middle rows).